An Improved Isomorphism Test for Bounded-Tree-Width Graphs

Martin Grohe
RWTH Aachen University, Aachen, Germany
grohe@informatik.rwth-aachen.de

Daniel Neuen
RWTH Aachen University, Aachen, Germany
neuen@informatik.rwth-aachen.de

Pascal Schweitzer
Technische Universität Kaiserslautern, Kaiserslautern, Germany
schweitzer@cs.uni-kl.de

Daniel Wiebking
RWTH Aachen University, Aachen, Germany
wiebking@informatik.rwth-aachen.de

Abstract

We give a new fpt algorithm testing isomorphism of n-vertex graphs of tree width k in time $2^{k \operatorname{polylog}(k)} \operatorname{poly}(n)$, improving the fpt algorithm due to Lokshtanov, Pilipczuk, Pilipczuk, and Saurabh (FOCS 2014), which runs in time $2^{\mathcal{O}(k^5 \log k)} \operatorname{poly}(n)$. Based on an improved version of the isomorphism-invariant graph decomposition technique introduced by Lokshtanov et al., we prove restrictions on the structure of the automorphism groups of graphs of tree width k. Our algorithm then makes heavy use of the group theoretic techniques introduced by Luks (JCSS 1982) in his isomorphism test for bounded degree graphs and Babai (STOC 2016) in his quasipolynomial isomorphism test. In fact, we even use Babai's algorithm as a black box in one place. We give a second algorithm which, at the price of a slightly worse run time $2^{\mathcal{O}(k^2 \log k)} \operatorname{poly}(n)$, avoids the use of Babai's algorithm and, more importantly, has the additional benefit that it can also be used as a canonization algorithm.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability, Mathematics of computing → Graph algorithms

Keywords and phrases graph isomorphism, graph canonization, tree width, decompositions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.67

Related Version A full version of the paper is available at `https://arxiv.org/abs/1803.06858`.

Funding Supported by the German Research Foundation DFG Koselleck Grant GR 1492/14-1

1 Introduction

Already early on in the beginning of research on the graph isomorphism problem (which asks for structural equivalence of two given input graphs) a close connection to the structure and study of the automorphism group of a graph was observed. For example, Mathon [11] argued that the isomorphism problem is polynomially equivalent to the task of computing a generating set for the automorphism group and also to computing the size of the automorphism group.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 67; pp. 67:1–67:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

With Luks's polynomial time isomorphism test for graphs of bounded degree [10], the striking usefulness of group theoretic techniques for isomorphism problems became apparent and they have been exploited ever since (e.g. [2, 12, 15, 13]). In his algorithm, Luks shows and uses that the automorphism group of a connected graph of bounded degree, after a vertex has been fixed, has a very restricted structure. More precisely, the group is in the class Γ_k of all groups whose composition factors are isomorphic to a subgroup of the symmetric group $\mathrm{Sym}(k)$.

Most recently, Babai's quasipolynomial time algorithm for general graph isomorphism [1] adds several novel techniques to tame and manage the groups that may appear as the automorphism group of the input graphs.

A second approach towards isomorphism testing is via decomposition techniques (e.g. [3, 5, 7]). These decompose the graph into smaller pieces while maintaining control of the complexity of the interplay between the pieces. When taking this route it is imperative to decompose the graph in an isomorphism-invariant fashion so as not to compare two graphs that have been decomposed in structurally different ways.

A prime example of this strategy is Bodlaender's isomorphism test [3] for graphs of bounded treewidth. Bodlaender's algorithm is a dynamic programming algorithm that takes into account all k-tuples of vertices that separate the graph, leading to a running time of $\mathcal{O}(n^{k+c})$ to test isomorphism of graphs of tree width at most k.

Only recently, Lokshtanov, Pilipczuk, Pilipczuk, and Saurabh [9] designed a fixed-parameter tractable isomorphism test for graphs of bounded tree width which has a running time of $2^{\mathcal{O}(k^5 \log k)}\,\mathrm{poly(n)}$. This algorithm first "improves" a given input graph G to a graph G^k by adding an edge between every pair of vertices between which more than k pairwise internally vertex disjoint paths exist. The improved graph G^k isomorphism-invariantly decomposes along clique separators into clique-separator free parts, which we will call *basic* throughout the paper. The decomposition can in fact be extended to an isomorphism-invariant tree decomposition into basic parts, as was shown in [4] to design a logspace isomorphism test for graphs of bounded tree width. For the basic parts, Lokshtanov et al. [9] show that, after fixing a vertex of sufficiently low degree, is it possible to compute an isomorphism-invariant tree decomposition whose bags have a size at most exponential in k and whose adhesion is at most $\mathcal{O}(k^3)$. They use this invariant decomposition to compute a canonical form essentially by a brute-force dynamic programming algorithm.

The problem of computing a canonical form is the task to compute, to a given input graph G, a graph G' isomorphic to G such that the output G' depends only on the isomorphism class of G and not on G itself.

The isomorphism problem reduces to the task of computing a canonical form: for two given input graphs we compute their canonical forms and check whether the canonical forms are equal (rather than isomorphic).

As far as we know, computing a canonical form could be algorithmically more difficult than testing isomorphism. It is usually not very difficult to turn combinatorial isomorphism tests into canonization algorithms, sometimes the algorithms are canonization algorithms in the first place. However, canonization based on group theoretic isomorphism tests is more challenging. For example, it is still open whether there is a graph canonization algorithm running in quasipolynomial time.

Our Results

Our main result is a new fpt algorithm testing isomorphism of graphs of bounded tree width.

▶ **Theorem 1.** *There is a graph isomorphism test running in time* $2^{k\,\mathrm{polylog}(k)}\,\mathrm{poly}(n)$*, where* n *is the size and* k *the minimum tree width of the input graphs.*

In the first part of the paper, we analyze the structure of the automorphism group of a graph G of tree width k. Following [9] and [4], we pursue a two-stage decomposition strategy for graphs of bounded tree width, where in the first step we decompose the improved graph along clique separators into basic parts. We observe that these basic parts are essential for understanding the automorphism groups. We show (Theorem 8) that with respect to a fixed vertex v of degree at most k, we can construct for each basic graph H an isomorphism-invariant tree decomposition of width at most $2^{\mathcal{O}(k \log k)}$ and adhesion at most $\mathcal{O}(k^2)$ where, in addition, each bag is equipped with a graph of small degree which is defined in an isomorphism-invariant way and gives us insight about the structure of the bag. In particular, using Luks's results [10], this also restricts the structure of the automorphism group.

Our construction is based on a similar construction of an isomorphism-invariant tree decomposition in [9]. Compared to that construction, we improve the adhesion (that is, the maximum size of intersections between adjacent bags of the decomposition) from $\mathcal{O}(k^3)$ to $\mathcal{O}(k^2)$. More importantly, we expand the decomposition by assigning a group and a graph to each bag.

Using these groups, we can prove that $\mathrm{Aut}(H)_v$ (the group of all automorphisms of H that keep the vertex v fixed) is a Γ_{k+1} group. This significantly restricts possible automorphism groups. Moreover, using the graph structure assigned to each bag, we can also compute the automorphism group of a graph of tree width k within the desired time bounds. The first, already nontrivial step towards computing the automorphism group, is a reduction from arbitrary graphs of tree width k to basic graphs. The second step reduces the problem of computing the automorphism group of a basic graph to the problem of computing the automorphism group of a structure that we call an *expanded d-ary tree*. In the reduction, the parameter d will be polynomially bounded in k. Then as the third step, we can apply a recent result [6] due to the first three authors that allows us to compute the automorphism groups of such expanded d-ary trees. This result is heavily based on techniques introduced by Babai [1] in his quasipolynomial isomorphism test. In fact, it even uses Babai's algorithm as a black box in one place.

We prove a second result that avoids the results of [6, 1] and even allows us to compute canonical forms, albeit at the price of an increased running time.

▶ **Theorem 2.** *There is a graph canonization algorithm running in time $2^{\mathcal{O}(k^2 \log k)} \mathrm{poly}(n)$, where n is the size and k the tree width of the input graph.*

Even though it does not employ Babai's new techniques, this algorithm still heavily depends on the group theoretic machinery. As argued above, the design of group theoretic canonization algorithms often requires extra work, and can be slightly technical, compared to the design of an isomorphism algorithm. Here, we need to combine the group theoretic canonization techniques going back to Babai and Luks [2] with graph decomposition techniques, which poses additional technical challenges and requires new canonization subroutines.

2 Preliminaries

Graphs. We use standard graph notation. All graphs $G = (V, E)$ considered are undirected finite simple graphs. We denote an edge $\{u, v\} \in E$ by uv. Let $U, W \subseteq V$ be subsets of vertices. We write $E(U, W)$ for the edges with one vertex in U and the other vertex from W, whereas $E(U)$ are the edges with both vertices in U. By $N(U)$, we denote the neighborhood of U, i.e., all vertices outside U that are adjacent to U. For the induced subgraph on U, we write $G[U]$, whereas $G - U$ is the induced subgraph on $V \setminus U$. A *rooted graph* is a

triple $G = (V, E, r)$ where $r \in V$ is the *root* of the graph. For two vertices $v, w \in V$ we denote by $\mathrm{dist}_G(v, w)$ the distance between v and w, i.e. the length of the shortest path from v to w. The *depth* of a rooted graph is the maximum distance from a vertex to the root, that is, $\mathrm{depth}(G) = \max_{v \in V} \mathrm{dist}_G(r, v)$. The *forward-degree* of a vertex $v \in V$ is $\mathrm{fdeg}(v) = |\{w \in N(v) \mid \mathrm{dist}(w, r) = \mathrm{dist}(v, r) + 1\}|$. Note that $|V| \leq (d+1)^{\mathrm{depth}(G)}$ where $d = \max_{v \in V} \mathrm{fdeg}(v)$ is the maximal forward-degree.

Separators. A pair (A, B) where $A \cup B = V(G)$ is called a *separation* if $E(A \setminus B, B \setminus A) = \emptyset$. In this case we call $A \cap B$ a *separator*. A separation (A, B) is an (L, R)-*separation* if $L \subseteq A$ and $R \subseteq B$ and in this case $A \cap B$ is called an (L, R)-*separator*. A separation (A, B) is a called a *clique separation* if $A \cap B$ is a clique and $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$. In this case we call $A \cap B$ a *clique separator*.

Tree Decompositions. A *tree decomposition* of a graph G is a pair (T, β), where T is a rooted tree and $\beta : V(T) \to \mathrm{Pow}(V(G))$ is a mapping into the power set of $V(G)$ such that:

1. for each vertex $v \in V(G)$, the set $\{t \in V(G) \mid v \in \beta(t)\}$ induces a nonempty and connected subtree of T, and
2. for each edge $e \in E(G)$, there exists $t \in V(T)$ such that $e \subseteq \beta(t)$.

Sets $\beta(t)$ for $t \in V(T)$ are called the *bags* of the decomposition, while sets $\beta(s) \cap \beta(t)$ for $st \in E(T)$ are called the *adhesions sets*. The *width* of a tree decomposition T is equal to its maximum bag size decremented by one, i.e. $\max_{t \in V(T)} |\beta(t)| - 1$. The *adhesion width* of T is equal to its maximum adhesion size, i.e. $\max_{st \in E(T)} |\beta(s) \cap \beta(t)|$. The *tree width* of a graph, denoted by $\mathrm{tw}(G)$, is equal to the minimum width of its tree decompositions.

A graph G is k-*degenerate* if every subgraph of G has a vertex with degree at most k. It is well known that every graph of tree width k is k-degenerate.

Groups. For a function $\phi \colon V \to V'$ and $v \in V$ we write v^ϕ for the image of v under ϕ, that is, $v^\phi = \phi(v)$. We write composition of functions from left to right, e.g, $v^{(\sigma\rho)} = (v^\sigma)^\rho = \rho(\sigma(v))$. By $[t]$ we denote the set of natural numbers from 1 to t. By $\mathrm{Sym}(V)$ we denote the symmetric group on a set V and we also write $\mathrm{Sym}(t)$ for $\mathrm{Sym}([t])$. We use upper case Greek letters $\Delta, \Phi, \Gamma, \Theta$ and Ψ for permutation groups.

Labeling cosets. A *labeling coset* of a set V is a set of bijective mappings $\tau\Delta$ where τ is a bijection from V to $[|V|]$ and Δ is a subgroup of $\mathrm{Sym}(|V|)$. By $\mathrm{Label}(V)$, we denote the labeling coset $\tau\,\mathrm{Sym}(|V|)$. We say that $\tau\Delta$ is a *labeling subcoset* of a labeling coset $\rho\Theta$, written $\tau\Delta \leq \rho\Theta$, if $\tau\Delta$ is a subset of $\rho\Theta$ and $\tau\Delta$ forms a labeling coset again. Sometimes we will choose a single symbol to denote a labeling coset $\tau\Delta$. For this will usually use the Greek letter Λ. Recall that Γ_k denotes the class of all finite groups whose composition factors are isomorphic to subgroups of $\mathrm{Sym}(k)$. Let $\widetilde{\Gamma}_k$ be the class of all labeling cosets $\Lambda = \tau\Delta$ such that $\Delta \in \Gamma_k$.

Orderings on sets of natural numbers. We extend the natural ordering of the natural numbers to finite sets of natural numbers. For two such sets M_1, M_2 we define $M_1 \prec M_2$ if $|M_1| < |M_2|$ or if $|M_1| = |M_2|$ and the smallest element of $M_1 \setminus M_2$ is smaller than the smallest element of $M_2 \setminus M_1$.

Isomorphisms. In this paper we will always define what the isomorphisms between our considered objects are. But this can also be done in a more general context. Let $\phi : V \to V'$. For a vector $(v_1, \dots, v_k)$ we define $(v_1, \dots, v_k)^\phi$ as $(v_1^\phi, \dots, v_k^\phi)$ inductively. Analogously, for a set we define $\{v_1, \dots, v_n\}^\phi$ as $\{v_1^\phi, \dots, v_n^\phi\}$. For a labeling coset $\Lambda \leq \mathrm{Label}(V)$ we write Λ^ϕ for $\phi^{-1}\Lambda$. In the paper we will introduce isomorphisms $\mathrm{Iso}(X, X')$ for various objects X and X'. Unless otherwise stated these are all $\phi : V \to V'$ such that $X^\phi = X'$ where we apply ϕ as previously defined. For example, the isomorphism between two graphs G and G' are all $\phi : V \to V'$ such that $G^\phi = G'$ which means that G has an edge uv, if and only if G' has the edge $u^\phi v^\phi$.

3 Clique separators and improved graphs

To perform isomorphism tests of graphs of bounded tree width, a crucial step in [9] is to deal with clique separators. For this step the concept of a k-improved graph is the key.

▶ **Definition 3** ([9]). The *k-improvement* of a graph G is the graph G^k obtained from G by connecting every pair of non-adjacent vertices v, w for which there are more than k pairwise internally vertex disjoint paths connecting v and w. We say that a graph G *is k-improved* when $G^k = G$.

A graph is *k-basic* if it is k-improved and does not have any separating cliques. In particular, a k-basic graph is connected.

We summarize several structural properties of G^k.

▶ **Lemma 4** ([9]). *Let G be a graph and $k \in \mathbb{N}$.*

1. *The k-improvement G^k is k-improved, i.e., $(G^k)^k = G^k$.*
2. *Every tree decomposition (T, β) of G of width at most k is also a tree decomposition of G^k.*
3. *There exists an algorithm that, given G and k, runs in $\mathcal{O}(k^2 n^3)$ time and either correctly concludes that $\mathrm{tw}(G) > k$, or computes G^k.*

Since the construction of G^k from G is isomorphism-invariant, the concept of the improved graph can be exploited for isomorphism testing and canonization. A k-basic graph has severe limitations concerning its structure as we explore in the following sections. In the canonization algorithm from [9] a result of Leimer [8] is exploited that says that every graph has a tree decomposition into clique-separator free parts, and the family of bags is isomorphism-invariant. While it is usually sufficient to work with an isomorphism-invariant set of bags (see [14]) we actually require an isomorphism invariant decomposition, which can indeed be obtained.

▶ **Theorem 5** ([8],[4]). *There is an algorithm that, given a connected graph G, computes a tree decomposition (T, β) of G, called clique separator decomposition, with the following properties.*

1. *For every $t \in V(T)$ the graph $G[\beta(t)]$ is clique-separator free (and in particular connected).*
2. *Each adhesion set of (T, β) is a clique.*
3. *$|V(T)| \in \mathcal{O}(|V(G)|)$.*
4. *For each bag $\beta(t)$ the adhesion sets of the children are all equal to $\beta(t)$ or the adhesion sets of the children are all distinct.*

The algorithm runs in polynomial time and the output of the algorithm is isomorphism-invariant (w.r.t. G).

4 Decomposing basic graphs

In this section, we shall construct bounded-width tree decompositions of k-basic graphs of tree width at most k. Crucially, these decompositions will be isomorphism-invariant after fixing one vertex of the graph. Our construction refines a similar construction of [9].

Let us define three parameters c_{S}, c_{M}, and c_{L} (small, medium and large) that depend on k: $c_{\mathrm{S}} \coloneqq 6(k+1) \in \mathcal{O}(k)$, $c_{\mathrm{M}} \coloneqq c_{\mathrm{S}}+c_{\mathrm{S}}(k+1) \in \mathcal{O}(k^2)$ and $c_{\mathrm{L}} \coloneqq c_{\mathrm{M}}+2(k+1)\binom{c_{\mathrm{M}}}{k+2}^2 \in 2^{\mathcal{O}(k \log k)}$.

The interpretation of these parameters is that c_{M} will bound the size of the adhesion sets and c_{L} will bound the bag size. The parameter c_{S} is used by the algorithm which in certain situations behaves differently depending on sets being larger than c_{S} or not.

The bound $c_{\mathrm{M}} \in \mathcal{O}(k^2)$ improves the corresponding bound $c_{\mathrm{M}} \in \mathcal{O}(k^3)$ in [9]. However, the more significant extension of the construction in [9] is that in addition to the tree decomposition we also construct both an isomorphism-invariant graph of bounded forward-degree and depth and an isomorphism-invariant Γ_{k+1}-group associated with each bag.

The *weight* of a set $S \subseteq V(G)$ with respect to a (weight) function $w : V(G) \to \mathbb{N}$ is $\sum_{v \in S} w(v)$. The *weight* of a separation (A, B) is the weight of its separator $A \cap B$. For sets $L, R \subseteq V(G)$, among all (L, R)-separations (A, B) of minimal weight there exists a unique separation with an inclusion minimal A. For this separation we call $A \cap B$ the *leftmost minimal separator* and denote it by $S_{L,R}(w)$. Moreover, we define $S_{L,R} = S_{L,R}(\mathbf{1})$ where $\mathbf{1}$ denotes the function that maps every vertex to 1.

For $U \subseteq V(G)$ we define a weight function $w_{U,k}$ such that $w_{U,k}(u) = k$ for all $u \in U$ and $w_{U,k}(v) = 1$ for all $v \in V \setminus U$. Given a weight function w, using Menger's theorem and the Ford-Fulkerson algorithm it is possible to compute $S_{L,R}(w)$. The following lemma generalizes Lemma 3.2 of [9]. Through this generalization we obtain the adhesion bound $\mathcal{O}(k^2)$ for our decomposition.

▶ **Lemma 6.** *Let G be a graph, let $S \subseteq V(G)$ be a subset of vertices, and let $\{T_i \subseteq V(G)\}_{i \in [t]}$ and $\{w_i : V(G) \to \mathbb{N}\}_{i \in [t]}$ be families where each T_i is a minimum weight (L_i, R_i)-separator with respect to w_i for some $L_i, R_i \subseteq S$. Let $w : V(G) \to \mathbb{N}$ be another weight function such that for all $i \in [t]$:*

1. *$w(v) = w_i(v)$ for all $v \in V(G) \setminus S$, and*
2. *$w(v) \geq w_i(v)$ for all $v \in V(G)$.*

Let $D \coloneqq S \cup \bigcup_{i \in [t]} T_i$. Suppose that Z is the vertex set of any connected component of $G - D$. Then $w(N(Z)) \leq w(S)$.

The lemma can be used to extend a set of vertices S that is not a clique separator to a set D in an isomorphism-invariant fashion while controlling the size of the adhesion sets of the components of $G - D$. It will be important for us that we can also extend a labeling coset of S to a labeling coset of D and furthermore construct a graph of bounded forward-degree and depth associated with D and S.

▶ **Lemma 7.** *Let $k \in \mathbb{N}$ and let G be a graph that is k-improved. Let $S \subseteq V(G)$ and let $\Lambda \leq \mathrm{Label}(S)$ be a labeling coset such that*

1. *$\emptyset \subsetneq S \subsetneq V(G)$,*
2. *$|S| \leq c_M$,*
3. *S is not a clique,*
4. *$G - S$ is connected,*
5. *$S = N_G(V(G) \setminus S)$, and*
6. *$\Lambda \in \widetilde{\Gamma}_{k+1}$.*

There is an algorithm that either correctly concludes that $tw(G) > k$, or finds a proper superset D of S and a labeling coset $\widehat{\Lambda} \leq \mathrm{Label}(D)$ and a connected rooted graph H with the following properties:

(A) $D \supsetneq S$,
(B) $|D| \leq c_L$,
(C) $\widehat{\Lambda} \in \widetilde{\Gamma}_{k+1}$,
(D) *if Z is the vertex set of any connected component of $G - D$, then $|N(Z)| \leq c_M$,*
(E) $D \subseteq V(H)$, $\operatorname{depth}(H) \leq k+3$ *and* $\operatorname{fdeg}(v) \in k^{\mathcal{O}(1)}$ *for all* $v \in V(H)$.

The algorithm runs in time $2^{\mathcal{O}(k \log k)}|V(G)|^{\mathcal{O}(1)}$ and the output $(D, \widehat{\Lambda}, H)$ is isomorphism-invariant (w.r.t. the input data G, S, Λ and k).

Here, the output of an algorithm $\mathcal{A}$ is isomorphism-invariant if all isomorphisms between two input data (G, S, Λ, k) and (G', S', Λ', k') extend to an isomorphism between the output $(D, \widehat{\Lambda}, H)$ and $(D', \widehat{\Lambda}', H')$ (an isomorphism between (G, S, Λ, k) and (G', S', Λ', k') is a mapping $\phi : V(G) \to V(G')$ such that $(G, S, \Lambda, k)^\phi = (G', S', \Lambda', k')$ where we apply ϕ as defined in the preliminaries).

Proof. We consider two cases depending on the size of S.

Case $|S| \leq c_S$: Let $\mathcal{I} := \{(\{x\}, \{y\}) \mid x, y \in S, x \neq y, xy \notin E(G)\}$ and let $D = S \cup \bigcup_{(L,R)\in\mathcal{I}} S_{L,R}(w_{L\cup R, k+1})$. We set $w := w_{S,k+1}$ and then we have the following for every vertex set Z of a connected component of $G - D$ by Lemma 6.

$$|N(Z)| \leq w(N(Z)) \overset{6}{\leq} w(S) \leq c_S(k+1) \leq c_M.$$

For every $xy \notin E(G)$ there is a $(\{x\}, \{y\})$-separator of size at most k disjoint from $\{x, y\}$, because G is k-improved. Thus $|D| \leq |S| + k|S|^2 \leq c_S + kc_S^2 \leq c_L$. Moreover, since $G - S$ is connected and $S = N_G(V(G) \setminus S)$, for all distinct $x, y \in S$ every minimum weight $(\{x\}, \{y\})$-separator contains a vertex that is not in S. It follows that $D \neq S$.

Case $c_S < |S| \leq c_M$: Let $\mathcal{I} := \{(L, R) \mid |L| = |R| \leq k+2, |S_{L,R}| \leq k+1\}$ and $D = S \cup \bigcup_{(L,R)\in\mathcal{I}} S_{L,R}$.

The properties of D follow from similar arguments as in the first case. The fact that $\mathcal{I}$ is nonempty follows from the existence of a *balanced* separation (for details see [9]). Next, we show how to find $\widehat{\Lambda}$ in both cases. To each $x \in D \setminus S$ we associate the set $A_x := \{(L, R) \in \mathcal{I} \mid x \in S_{L,R}\}$. Two vertices x and y occur in exactly the same separators if $A_x = A_y$. In this case we call them equivalent and write $x \equiv y$. Let $A_1, \ldots, A_t \subseteq D \setminus S$ be the equivalence classes of "$\equiv$". Since each x is contained in some separator of size at most $k+1$ we conclude that the size of each A_i is at most $k+1$.

For each labeling $\lambda \in \operatorname{Label}(S)$ we choose an extension $\widehat{\lambda} : D \to \{1, \ldots, |D|\}$ such that $\widehat{\lambda}|_S = \lambda$ and for $x, y \in D \setminus S$ we have $x^{\widehat{\lambda}} < y^{\widehat{\lambda}}$ if $A_x^\lambda \prec A_y^\lambda$. (Recall that $\prec$ is the linear order of subsets of $\mathbb{N}$ as defined in the preliminaries). Inside each equivalence class A_i, the ordering is chosen arbitrarily. Define $\widehat{\Lambda} := (\{\mathrm{id}_S\} \times \operatorname{Sym}(A_1) \times \ldots \times \operatorname{Sym}(A_t)) \cdot \{\widehat{\lambda} \mid \lambda \in \Lambda\} \leq \operatorname{Label}(D)$. By construction the coset $\widehat{\Lambda}$ does not depend on the choices of the extensions $\widehat{\lambda}$. Since $|A_i| \leq k+1$ for all $1 \leq i \leq t$ we conclude that $\widehat{\Lambda} \in \widetilde{\Gamma}_{k+1}$, as desired.

It remains to explain how to efficiently compute $\widehat{\Lambda}$. For this we simply remark that it suffices to use a set of extensions $M \subseteq \Lambda$ such that Λ is the smallest coset containing all elements of M (i.e., we can use a coset analogue of a generating set). We conclude that $\widehat{\Lambda}$ can be computed in polynomial time in the size of $\mathcal{I}$.

Last but not least, we show how to construct the graph H. The Case $|S| \leq c_S$ is easy to handle. In this case we define H as the complete graph on the set $D \cup \{r\}$ where r is some new vertex, which becomes the root of H. The forward-degree of r is bounded by $|D|$ which in turn is bounded by $k^{\mathcal{O}(1)}$. We consider the Case $c_S < |S| \leq c_M$. We define $V(H) := \{(L, R) \mid L, R \subseteq S, |L| = |R| \leq k+2\} \cup D$. Clearly, we have $\mathcal{I} \subseteq V(H)$. For the root we choose $(\emptyset, \emptyset) \in V(H)$. We define the edges $E(H) := \{(L, R)(L', R') \mid L \subseteq L', R \subseteq R', |L| +$

$1 = |R|+1 = |L'| = |R'|\} \cup \{x(L,R) \mid x \in D, x \in S_{L,R}, (L,R) \in \mathcal{I}\}$. Since for each pair (L,R) there are at most $|S|^2$ different extensions (L',R') with $|L|+1 = |R|+1 = |L'| = |R'|$ and since each separator $S_{L,R}$ contains at most $k+1$ vertices, we conclude that the forward-degree of each vertex in H is bounded by $|S|^2+k+1 \in k^{\mathcal{O}(1)}$. The depth of H is at most $k+3$. ◀

A *labeled tree decomposition* (T,β,α,η) is a 4-tuple where (T,β) is a tree decomposition and α is a function that maps each $t \in V(T)$ to a labeling coset $\alpha(t) \leq \text{Label}(\beta(t))$ and η is a function that maps each $t \in V(T)$ to a graph $\eta(t)$.

The previous lemma can be used as a recursive tool to compute our desired isomorphism-invariant labeled tree decomposition.

▶ **Theorem 8.** *Let $k \in \mathbb{N}$ and let G be a k-basic graph and let v be a vertex of degree at most k. There is an algorithm that either correctly concludes that $\text{tw}(G) > k$, or computes a labeled tree decomposition (T,β,α,η) with the following properties.*

1. *the width of (T,β) is bounded by c_L,*
2. *the adhesion width of (T,β) is bounded by c_M,*
3. *the degree of T is bounded by kc_L^2 and the number of children of t with common adhesion set is bounded by k for each $t \in V(T)$,*
4. *$|V(T)|$ is bounded by $\mathcal{O}(|V(G)|)$,*
5. *for each bag $\beta(t)$ the adhesion sets of the children are all equal to $\beta(t)$ or the adhesion sets of the children are all distinct. In the former case the bag size is bounded by c_M,*
6. *for each $t \in V(T)$ the graph $\eta(t) = H_t$ is a connected rooted graph such that $\beta(t) \cup \beta(t)^2 \subseteq V(H_t)$ and for each adhesion set S there is a corresponding vertex $S \in V(H_t)$, $\text{depth}(H_t) \in \mathcal{O}(k)$ and $\text{fdeg}(v) \in k^{\mathcal{O}(1)}$ for all $v \in V(H_t)$, and*
7. *$\alpha(t) \in \widetilde{\Gamma}_{k+1}$.*

The algorithm runs in time $2^{\mathcal{O}(k^2 \log k)}|V(G)|^{\mathcal{O}(1)}$ and the output (T,β,α,η) of the algorithm is isomorphism invariant (w.r.t. G, v and k). Furthermore, if we drop Property 7 as a requirement, the triple (T,β,η) can be computed in time $2^{\mathcal{O}(k \log k)}|V(G)|^{\mathcal{O}(1)}$.

Here, the output of an algorithm $\mathcal{A}$ is isomorphism-invariant, if all isomorphisms between two input data extend to an isomorphism between the output. More precisely, an isomorphism $\phi \in \text{Iso}((G,v,k),(G',v',k'))$ extends to an isomorphism between (T,β,α,η) and $(T',\beta',\alpha',\eta')$ if there is a bijection between the tree decompositions $\phi_T : V(T) \to V(T')$ and for each node $t \in V(T)$ a bijection between the vertices of graphs $\phi_t : V(\eta(t)) \to V(\eta'(\phi_T(t)))$ which extends ϕ, i.e. $\phi_t(x) = x^\phi$ for all $x \in \beta(t) \cup \beta(t)^2 \cup 2^{\beta(t)}$ where we naturally apply ϕ as defined in the preliminaries. Furthermore, these extensions define an isomorphism between the output data, i.e. for all nodes $t \in V(T)$ we have that $\beta(t)^\phi = \beta'(\phi_T(t))$, $\alpha(t)^\phi = \alpha'(\phi_T(t))$ and $\eta(t)^{\phi_t} = \eta'(\phi_T(t))$.

▶ Remark. We later use the isomorphism-invariance of the labeled tree decomposition (T,β,α,η) from the previous theorem in more detail. Let $t \in V(T)$ be a non-root node and let $S \subseteq V(G)$ be the adhesion set to the parent node of t and let $I_t = (T_t,\beta_t,\alpha_t,\eta_t)$ be the decomposition of the subtree rooted at t and G_t the graph corresponding to I_t. Then η_t is isomorphism-invariant w.r.t. T_t, β_t, G_t and S.

5 Coset-Hypergraph-Isomorphism

After having computed isomorphism-invariant tree decompositions in the previous sections we now want to compute the set of isomorphisms from one graph to another in a bottom up fashion. Let G_1, G_2 be the two input graphs and suppose we are given isomorphism-invariant

tree decompositions (T_1, β_1) and (T_2, β_2). For a node $t \in V(T_i)$ we let $(G_i)_t$ be the graph induced by the union of all bags contained in the subtree rooted at t. The basic idea is to compute for all pairs $t \in V(T_1), t' \in V(T_2)$ the set of isomorphisms from $(G_1)_t$ to $(G_2)_{t'}$ (in addition the isomorphisms shall also respect the underlying tree decomposition) in a bottom up fashion.

The purpose of this section is to give an algorithm that solves this problem at a given bag (assuming we have already solved the problem for all pairs of children of t and t'). Let us first give some intuition for this task. Suppose we are looking for all bijections from $\beta_1(t)$ to $\beta_2(t')$ that can be extended to an isomorphism from $(G_1)_t$ to $(G_2)_{t'}$. Let $t_1, \dots, t_\ell$ be the children of t and $t'_1, \dots, t'_\ell$ the children of t'. Then we essentially have to solve the following two problems. First, we have to respect the edges appearing in the bags $\beta_1(t)$ and $\beta_2(t')$. But also, every adhesion set $\beta(t) \cap \beta(t_i)$ has to be mapped to another adhesion set $\beta(t') \cap \beta(t'_j)$ in such a way that the corresponding bijection (between the adhesion sets) extends to an isomorphism from $(G_1)_{t_i}$ to $(G_2)_{t'_j}$. In order to solve this problem we first consider the case in which the adhesion sets are all distinct and define the following abstraction.

An instance of *coset-hypergraph-isomorphism* is an 8-tuple $\mathcal{I} = (V_1, V_2, \mathcal{S}_1, \mathcal{S}_2, \chi_1, \chi_2, \mathcal{F}, \mathfrak{f})$ such that

1. $\mathcal{S}_i \subseteq \mathrm{Pow}(V_i)$,
2. $\chi_i \colon \mathcal{S}_i \to \mathbb{N}$ is a coloring,
3. $\mathcal{F} = \{\Theta_S \leq \mathrm{Sym}(S) \mid S \in \mathcal{S}_1\}$, and
4. $\mathfrak{f} = \{\tau_{S_1,S_2} \colon S_1 \to S_2 \mid S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2 \text{ such that } \chi_1(S_1) = \chi_2(S_2)\}$ such that
   a. every $\tau_{S_1,S_2} \in \mathfrak{f}$ is bijective, and
   b. for every color $i \in \mathbb{N}$ and every $S_1, S'_1 \in \chi_1^{-1}(i)$ and $S_2, S'_2 \in \chi_2^{-1}(i)$ and $\theta \in \Theta_{S_1}, \theta' \in \Theta_{S'_1}$ it holds that

$$\theta \tau_{S_1,S_2} (\tau_{S'_1,S_2})^{-1} \theta' \tau_{S'_1,S'_2} \in \Theta_{S_1} \tau_{S_1,S'_2}. \tag{N}$$

The instance is *solvable* if there is a bijective mapping $\phi \colon V_1 \to V_2$ such that

1. $S \in \mathcal{S}_1$ if and only if $S^\phi \in \mathcal{S}_2$ for all $S \in \mathrm{Pow}(V_1)$,
2. $\chi_1(S) = \chi_2(S^\phi)$ for all $S \in \mathcal{S}_1$, and
3. for every $S \in \mathcal{S}_1$ it holds that $\phi|_S \in \Theta_S \tau_{S,S^\phi}$.

In this case we call ϕ an *isomorphism* of the instance $\mathcal{I}$. Moreover, let $\mathrm{Iso}(\mathcal{I})$ be the set of all isomorphisms of $\mathcal{I}$. Observe that property (N) describes a consistency condition: if we can use σ_1 to map S_1 to S_2, σ_2 to map S'_1 to S_2, and σ_3 to map S'_1 to S'_2, then the mapping $\sigma_1 \sigma_2^{-1} \sigma_3$ can be used to map S_1 to S'_2. As a result the set of all isomorphisms of the instance $\mathcal{I}$ forms a coset, that is $\mathrm{Iso}(\mathcal{I}) = \Theta\phi$ for some $\Theta \leq \mathrm{Sym}(V_1)$ and $\phi \in \mathrm{Iso}(\mathcal{I})$.

In the application in the main recursive algorithm, the sets $V_i = \beta(t_i)$, the hyperedges $\mathcal{S}_i$ are the adhesion sets of t_i (and we will also encode the edges appearing in the bag in this way), and the cosets $\Theta_{S_1} \tau_{S_1,S_2}$ tell us which mappings between the adhesion sets S_1 and S_2 extend to an isomorphism between the corresponding subgraphs. The colorings χ_1 and χ_2 are used to indicate which subgraphs can not be mapped to each other (and also to distinguish between the adhesion sets and edges of the bags which will both appear in the set of hyperedges).

The next Lemma gives us one of the central subroutines for our recursive algorithm.

▶ **Lemma 9.** *Let $\mathcal{I} = (V_1, V_2, \mathcal{S}_1, \mathcal{S}_2, \chi_1, \chi_2, \mathcal{F}, \mathfrak{f})$ be an instance of coset-hypergraph-isomorphism. Moreover, suppose there are isomorphism-invariant rooted graphs $H_1 = (W_1, E_1, r_1)$ and $H_2 = (W_2, E_2, r_2)$ such that*

1. *$V_i \cup \mathcal{S}_i \subseteq W_i$,*
2. *$\mathrm{fdeg}(w) \leq d$ for all $w \in W_i$,*

3. $\operatorname{depth}(H_i) \leq h$, *and*
4. $|S| \leq d$ *for all* $S \in \mathcal{S}_i$

for all $i \in \{1, 2\}$. *Then a representation for the set* $\operatorname{Iso}(\mathcal{I})$ *can be computed in time* $2^{\mathcal{O}(h \cdot (\log d)^c)}$ *for some constant* c.

Here, isomorphism-invariant means that for every isomorphism $\phi \in \operatorname{Iso}(\mathcal{I})$ there is an isomorphism ϕ_H from H_1 to H_2 such that $v^\phi = v^{\phi_H}$ for all $v \in V_1$ and $\{v^\phi \mid v \in S\} = S^{\phi_H}$ for all $S \in \mathcal{S}_1$.

The proof of this lemma is based on the following theorem.

▶ **Theorem 10** ([6]). *Let* $\mathfrak{x}, \mathfrak{y} \colon V \to \Sigma$ *be two strings, let* $\Gamma \leq \operatorname{Sym}(V)$ *be a* Γ_d*-group and* $\gamma \in \operatorname{Sym}(V)$. *Then one can compute a representation of the set of all permutations* $\gamma' \in \Gamma\gamma$ *mapping* $\mathfrak{x}$ *to* $\mathfrak{y}$ *in time* $n^{\mathcal{O}((\log d)^c)}$ *for some constant* c *where* $n = |V|$.

Actually, we shall only need the following corollary. A rooted tree $T = (V, E, r)$ is *d-ary* if every node has at most d children. An expanded rooted tree is a tuple (T, C) where $T = (V, E, r)$ is a rooted tree and $C \colon L(T)^2 \to \operatorname{rg}(C)$ is a coloring of pairs of leaves of T ($L(T)$ denotes the set of leaves of T). Isomorphisms between expanded trees (T, C) and (T, C') are required to respect the colorings C and C'.

▶ **Corollary 11.** *Let* (T, C) *and* (T', C') *be two expanded* d*-ary trees and let* $\Gamma \leq \operatorname{Aut}(T)$ *and* $\gamma \in \operatorname{Iso}(T, T')$. *Then one can compute a representation of the set* $\{\phi \in \Gamma\gamma \mid (T, C)^\phi = (T', C')\}$ *in time* $n^{\mathcal{O}((\log d)^c)}$ *for some constant* c.

Proof sketch of Lemma 9. The proof essentially proceeds in three steps. First, the two graphs H_i ($i = 1, 2$) are extended by adding a vertex (S_i, v) for every $S_i \in \mathcal{S}_i$, $v \in S_i$ which is connected to the vertex S_i. Moreover, we compute the tree unfoldings T_i of both extended graphs.

Then, in the second step, we compute the set of isomorphisms γ from T_1 to T_2 such that for every $S \in \mathcal{S}_1$ the set $\{S\} \times S$ is mapped to $(\{S\} \times S)^\gamma$ according to the restrictions given by the cosets from the instance $\mathcal{I}$. This is possible since all the sets may be mapped independently of each other. In particular, at this point, identical elements $v \in V_1$ appearing in different sets $S, S' \in \mathcal{S}_1$ may be mapped to different elements $v', v'' \in V_2$.

To resolve this problem we use the coloring on the pairs of leaves of T_1 and T_2 to encode which elements in the tree unfolding correspond to identical elements in the original graph. The set of isomorphisms respecting these additional constraints can be computed using Corollary 11. ◀

Looking at the properties of the tree decompositions computed in Theorem 5 and 8 we have for every node t that either the adhesion sets to the children are all equal or they are all distinct. Up to this point we have only considered the problem that all adhesion sets are distinct (i.e. the coset-hypergraph-isomorphism problem). Next we consider the case that all adhesion sets are equal. Towards this end we define the following variant.

An instance of *multiple-colored-coset-isomorphism* is a 6-tuple $\mathcal{I} = (V_1, V_2, \chi_1, \chi_2, \mathcal{F}, \mathfrak{f})$ such that

1. $\chi_i \colon [t] \to \mathbb{N}$ is a coloring,
2. $\mathcal{F} = \{\Theta_i \leq \operatorname{Sym}(V) \mid i \in [t]\}$, and
3. $\mathfrak{f} = \{\tau_{i,j} \colon V_1 \to V_2 \mid i, j \in [t] \text{ such that } \chi_1(i) = \chi_2(j)\}$ such that
   a. every $\tau_{i,j} \in \mathfrak{f}$ is bijective, and
   b. for every color $i \in \mathbb{N}$ and every $j_1, j_1' \in \chi_1^{-1}(i)$ and $j_2, j_2' \in \chi_2^{-1}(i)$ and $\theta \in \Theta_{j_1}, \theta' \in \Theta_{j_1'}$ it holds that

$$\theta \tau_{j_1, j_2} \tau_{j_1', j_2}^{-1} \theta' \tau_{j_1', j_2'} \in \Theta_{j_1} \tau_{j_1, j_2'}. \tag{N2}$$

The instance is *solvable* if there is a bijective mapping $\phi\colon V_1 \to V_2$ and a $\pi \in \mathrm{Sym}(t)$ such that

1. $\chi_1(i) = \chi_2(\pi(i))$ for all $i \in [t]$, and
2. for every $i \in [t]$ it holds that $\phi \in \Theta_i \tau_{i,\pi(i)}$.

The set $\mathrm{Iso}(\mathcal{I})$ is defined analogously.

▶ **Lemma 12.** *Let $\mathcal{I} = (V_1, V_2, \chi_1, \chi_2, \mathcal{F}, \mathfrak{f})$ be an instance of multiple-colored-coset-isomorphism. Then a representation for the set $\mathrm{Iso}(\mathcal{I})$ can be computed in time*

$$\min((|V_1|!|\mathcal{F}|)^{\mathcal{O}(1)}, |\mathcal{F}|!^{\mathcal{O}(1)} 2^{\mathcal{O}((\log |V_1|)^c)}).$$

6 The isomorphism and the canonization algorithm

6.1 The isomorphism test

Before describing the main algorithm we need to state the following auxiliary lemma.

▶ **Lemma 13.** *Let $G = (D, E)$ be a graph of tree width at most k and let H be a rooted graph such that $D \subseteq V(H)$. Then, one can compute an isomorphism-invariant rooted graph H' such that*

1. *H is an induced subgraph of H',*
2. *$\mathrm{fdeg}_{H'}(w) \leq \max\{d, k+1\} + 1$ for all $w \in V(H')$ where $d = \max_{w \in V(H)} \mathrm{fdeg}_H(w)$,*
3. *$\mathrm{depth}(H') \leq \mathrm{depth}(H) + k + 2$, and*
4. *for every clique $C \subseteq D$ there is a corresponding vertex $C \in V(H')$*

in time $2^{\mathcal{O}(k)} \cdot |V(H)|^{\mathcal{O}(1)}$.

Here, isomorphism invariant means that every isomorphism $\phi \in \mathrm{Iso}(H_1, H_2)$, which naturally restricts to an isomorphism from G_1 to G_2, can be extended to an isomorphism from H_1' to H_2'.

▶ **Theorem 14.** *Let $k \in \mathbb{N}$ and let G_1, G_2 be connected graphs. There is an algorithm that either correctly concludes that $\mathrm{tw}(G_1) > k$, or computes the set of isomorphisms $\mathrm{Iso}(G_1, G_2)$ in time $2^{\mathcal{O}(k(\log k)^c)} |V(G)|^{\mathcal{O}(1)}$ for some constant c.*

Proof sketch. The first step is to decompose the given graphs. Notice that the decomposition techniques in Theorem 8 can only be applied to k-basic graphs, i.e. k-improved and clique-separator free. Therefore, we proceed as follows. First, we consider the k-improvement G_1^k and G_2^k of G_1 and G_2, respectively. We build the clique-separator decompositions of the k-improvements using Theorem 5. Secondly, we refine each (k-basic) bag of the decomposition by constructing a labeled decomposition for each bag using Theorem 8. (In fact, we need to fix a vertex in order to apply Theorem 8. For this reason we are just able to construct a family of decompositions in each k-basic bag. But it turns out, that we are also able to handle this. Also note that we use the version of Theorem 8 that runs in time $2^{\mathcal{O}(k \log k)} |V(G)|^{\mathcal{O}(1)}$ and omits the labeling cosets $\alpha(t)$.) The crucial point is that in our final decomposition the size of each bag is bounded by $c_{\mathrm{L}} \in 2^{\mathcal{O}(k \log k)}$ and more importantly each bag is assigned to a graph that restricts possible automorphisms of the corresponding graph structure. More precisely each bag $\beta(t)$ is assigned to a graph $\eta(t)$ of bounded degree such that $V(\eta(t))$ contains the vertices of $\beta(t)$, the edges of $\beta(t)$ and also the adhesion sets of t. In fact, we also need to embed the clique separators of the outer decomposition into the graph $\eta(t)$. However, this can be achieved using Lemma 13.

From now on, we start to compute isomorphisms preserving the decompositions T_1 and T_2 in a bottom up fashion. (In order to compute edge-preserving bijections, we need to consider

the edges of G_1 and G_2 rather than the "improved" edges in G_1^k and G_2^k). The decomposition of both graphs have a root bags denoted t_1 and t_2, respectively. Both roots have children $t_{11}, \dots, t_{1\ell}$ and $t_{21}, \dots, t_{2\ell}$. By T_{ji} we denote the decomposition rooted at t_{ji} and consisting of all bags that are descendants of t_{ji}. By using a dynamic programming approach we assume that the isomorphisms from T_{1i} to $T_{2i'}$ are already computed. To compute the isomorphisms from T_1 to T_2 we use our subroutine, namely coset-hypergraph-isomorphism, which is defined and algorithmically solved in Section 5. For a moment, assume that all adhesion sets of t_1 (and t_2, respectively) are distinct. We restrict the computed isomorphisms from T_{1i} to $T_{2i'}$ to their corresponding adhesions sets of t_1 and t_2, respectively. These restrictions are used to define $\mathcal{F}$ and $\mathfrak{f}$. The edge relation can also be encoded in $\mathcal{F}$ and $\mathfrak{f}$. More precisely, we add each edge $uv \in E(\beta_1(t_1))$ and $u'v' \in E(\beta_2(t_2))$ to $\mathcal{S}_1$ and $\mathcal{S}_2$, and define $\Theta_{uv}\tau_{uv,u'v'}$ as the set of bijections mapping uv to $u'v'$. The colorings χ_1 and χ_2 help to define the right instance, e.g. they distinguish the edges from the adhesion sets in $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Finally, we solve the instance $(\beta_1(t_1), \beta_2(t_2), \mathcal{S}_1, \mathcal{S}_2, \chi_1, \chi_2, \mathcal{F}, \mathfrak{f})$ with the algorithm from Lemma 9. In case all adhesion sets are equal we simply use Lemma 12 instead. ◀

▶ Remark. The degree of the polylogarithmic term (i.e. the constant c) in the running time of the previous theorem is related to the corresponding constant in Babai's quasipolynomial time isomorphism test [1]. Since the constant that Babai's algorithm achieves is not specified in [1], we also do not specify the constant.

6.2 Canonization

We briefly describe how to adapt our techniques to obtain an algorithm which computes a canonization for a given graph of tree width of at most k. Since Babai's quasipolynomial time algorithm [1] only tests isomorphism of two given input graphs and can not be used for canonization purposes, we need to replace the methods introduced in Section 5. To achieve this we still use group theoretic techniques, but compared to isomorphism the machinery is quite lengthy and technical.

▶ **Theorem 15.** *There is a graph canonization algorithm running in time* $2^{\mathcal{O}(k^2 \log k)} \operatorname{poly}(n)$*, where* n *is the size and* k *the tree width of the input graph.*

Proof sketch. The basic approach for the canonization algorithm is very similar to the approach presented in Theorem 14. One of the main differences is how the algorithm gets its insight into the structure of the bags of the decomposition of the k-basic graphs. In the isomorphism algorithm the graph $\eta(t)$ serves as a tool to exploit the structure of each bag t. For the canonization algorithm we instead use the fact that each bag can be guarded with a labeling coset $\alpha(t)$ of bounded composition-width.

We then define the corresponding variant of the coset-hypergraph-isomorphism problem suited for canonization. For the algorithm we assume that, instead of graphs of small degree and depth, we get a labeling coset of composition width at most $k+1$. This problem can then be solved in time $n^{\mathcal{O}(k)}$ using the group theoretic techniques from [2, 12, 15] where n denotes the number of vertices (i.e. the number of vertices in a single bag in the application in the main canonization algorithm). Since the bag size is bounded by $2^{\mathcal{O}(k \log k)}$ this gives the overall running time of $2^{\mathcal{O}(k^2 \log k)} \operatorname{poly}(n)$. ◀

References

1. László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. doi:10.1145/2897518.2897542.
2. László Babai and Eugene M. Luks. Canonical labeling of graphs. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 171–183. ACM, 1983. doi:10.1145/800061.808746.
3. Hans L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11(4):631–643, 1990. doi:10.1016/0196-6774(90)90013-5.
4. Michael Elberfeld and Pascal Schweitzer. Canonizing graphs of bounded tree width in logspace. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 32:1–32:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. URL: http://www.dagstuhl.de/dagpub/978-3-95977-001-9, doi:10.4230/LIPIcs.STACS.2016.32.
5. Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM J. Comput.*, 44(1):114–159, 2015. doi:10.1137/120892234.
6. Martin Grohe, Daniel Neuen, and Pascal Schweitzer. A faster isomorphism test for graphs of small degree. *CoRR*, abs/1802.04659, 2018. arXiv:1802.04659.
7. Martin Grohe and Pascal Schweitzer. Isomorphism testing for graphs of bounded rank width. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1010–1029. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.66.
8. Hanns-Georg Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1-3):99–123, 1993. doi:10.1016/0012-365X(93)90510-Z.
9. Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM J. Comput.*, 46(1):161–189, 2017. doi:10.1137/140999980.
10. Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.*, 25(1):42–65, 1982. doi:10.1016/0022-0000(82)90009-5.
11. Rudolf Mathon. A note on the graph isomorphism counting problem. *Inf. Process. Lett.*, 8(3):131–132, 1979. doi:10.1016/0020-0190(79)90004-8.
12. Gary L. Miller. Isomorphism testing and canonical forms for k-contractable graphs (A generalization of bounded valence and bounded genus). In Marek Karpinski, editor, *Fundamentals of Computation Theory, Proceedings of the 1983 International FCT-Conference, Borgholm, Sweden, August 21-27, 1983*, volume 158 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 1983. doi:10.1007/3-540-12689-9_114.
13. Daniel Neuen. Graph isomorphism for unit square graphs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 70:1–70:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. URL: http://drops.dagstuhl.de/opus/portals/lipics/index.php?semnr=16013, doi:10.4230/LIPIcs.ESA.2016.70.
14. Yota Otachi and Pascal Schweitzer. Reduction techniques for graph isomorphism in the context of width parameters. In R. Ravi and Inge Li Gørtz, editors, *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July*

2-4, 2014. Proceedings, volume 8503 of *Lecture Notes in Computer Science*, pages 368–379. Springer, 2014. doi:10.1007/978-3-319-08404-6_32.

15 Ilia N. Ponomarenko. The isomorphism problem for classes of graphs closed under contraction. *Journal of Mathematical Sciences*, 55(2):1621–1643, 1991.